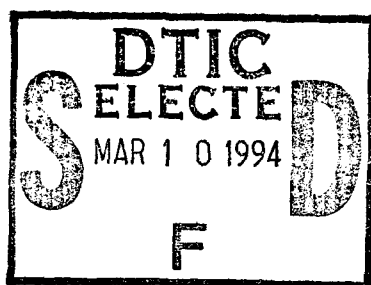IDA PAPER P-2975

ECONOMIC FOUNDATIONS FOR PRICING SOFTWARE REUSE REPOSITORIES

Thomas P. Frazier, *Project Leader*

Elizabeth K. Bailey
Bruce N. Angier

DTIC
SELECTED
MAR 1 0 1994
F

September 1994

Prepared for
Office of the Assistant Secretary of Defense
(Command, Control, Communications and Intelligence)

19950308 140

INSTITUTE FOR DEFENSE ANALYSES
1801 N. Beauregard Street, Alexandria, Virginia 22311-1772

IDA PAPER P-2975

ECONOMIC FOUNDATIONS FOR PRICING SOFTWARE REUSE REPOSITORIES

Thomas P. Frazier, *Project Leader*

Elizabeth K. Bailey
Bruce N. Angier

Accesion For	
NTIS CRA&I	☑
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

September 1994

IDA

INSTITUTE FOR DEFENSE ANALYSES

PREFACE

This paper was prepared by the Institute for Defense Analyses (IDA) for the Office of the Assistant Secretary of Defense (Command, Control, Communications and Intelligence) under a task entitled "Software Reuse Repositories." The objective of the task was to examine the options for government software repositories under a fee-for-service structure and to assess their economic impact.

This work was reviewed within IDA by An-Jen Tai, Michael S. Nash, and Stephen K. Welman.

CONTENTS

FIGURES

TABLES

EXECUTIVE SUMMARY

The Department of Defense envisions assembling its future software programs component by component instead of line by line. A key part of DoD plans for instituting this software development and maintenance approach involves building, populating, and operating repositories of reusable software components. These repositories will serve as intermediaries between suppliers and consumers of software components.

In this paper, we argue that the economic role that repositories play is as an intermediary to reduce the risk and uncertainty for both the contributor and subscriber of software products and services. This reduction of risk and uncertainty has the effect of reducing the costs to the contributors and subscribers and ultimately to the government.

The original question we were asked to address was: What prices should DoD software repositories charge for their products under a fee-for-service structure? That is, how can the pricing mechanism be employed to strike a balance between encouraging the diffusion of reusable assets and recovering the cost of operating the repository? In our view, that question is secondary to a more fundamental question regarding incentives. Studies have shown that in order to induce systematic software reuse, software must be made reusable by making it "generalizable" (i.e., suitable for reuse). Generalizing software assets can carry a significant cost. Unless the asset creators are rewarded for making their software assets reusable and contributing that software to the repository, we believe the most likely outcome will be repositories full of unusable and unused software assets regardless of what price might be charged. Thus, the fundamental question to be addressed is: Who has any incentive to place truly reusable assets in a repository? The underlying goal then becomes how to strike a balance between encouraging the diffusion of reusable assets, recovering the cost of operating the repository, and *rewarding the asset creators for their efforts*.

An approach emphasizing greater private property rights in software would create incentives to the software creators to supply reusable assets. Under this regime, the repository would function as a market intermediary first serving to bring contributors and subscribers together and then stepping aside. Prices, data rights, maintenance arrangements, training, and other related issues would be determined by contributors and

subscribers dealing with each other directly rather than through the repository, as is the current scheme.

Regardless of the property rights issue, the DoD has a significant role to play in software reuse. The delineation and enforcement of domain models and architectures for domains that are specific to DoD (e.g., electronic warfare, command and control) are obvious examples of functions over which DoD has good reason to retain ownership and control. In case the government continues to be directly involved with repository contributors and subscribers, the paper makes recommendations for pricing of repository products and services.

I. INTRODUCTION

A. SOFTWARE REUSE IN THE DEPARTMENT OF DEFENSE (DOD)

In the DoD's vision of the future, most software systems will be assembled from reusable components, allowing software development to proceed more quickly and more predictably and resulting in software that is more reliable than that available from the built-from-scratch methods in use today. Under the reuse paradigm, the software developer would select from libraries or repositories of software components and build the program component by component instead of the traditional line-by-line method. Examples of a component would be a standard mathematical or statistical routine, a radar-jamming package for the defensive avionics of an aircraft, or a tax-withholding package in a payroll system. Different components are subject to different levels or degrees of reuse. Some components can be reused verbatim while others must be adapted before being incorporated in the new program.

Certain forms of reuse are already a prevalent aspect of any typical software development effort. High-order languages, editor macros, device drivers, and documentation templates are all examples of reuse on some level. For the purpose of this paper, however, we define "reuse" as replacing all or part of the conventional development software artifacts with existing assets, or with derivatives for outputs from existing assets.[1]

The payoff of software reuse is expected to come in the form of increased productivity (i.e., the amount of output per unit of labor effort) by reducing the time and effort needed to develop software [2 through 4]. Reuse should also enhance labor productivity in the maintenance phase of the software program life cycle [5] by increasing software quality (measured in term of errors) due to multiple testing and error-removal opportunities. The smaller the number of errors, the lower the number of corrective maintenance actions incurred as a result of reuse [6 and 7].

---

[1] Software reuse is defined by the "DoD Software Technology Strategy" to be "the deliberate exploitation of existing software components and related assets to facilitate the development of new software systems" [1, p. GL-18]. This definition covers systematic reuse and excludes opportunistic, or ad hoc, reuse.

The DoD's vision of assembling future software programs component by component will require investment to build, populate, and operate repositories of reusable software components.[2] Repositories will serve as intermediaries between suppliers and consumers of software components. The "DoD Software Technology Strategy" [1] defines a repository to be: [1, p. GL-18].

> "the place where the software artifacts are kept in their on-line form. Its functions are modeled after those of a librarian; it is the gatekeeper of project products. Artifacts are typically created and modified outside the realm of the repository."

The potential contributors to the repository (software suppliers) consist of commercial vendors, government contractors, government in-house developers, and universities. The potential subscribers to the repository (software consumers) consist of those same groups. Contributors and subscribers will interact through the repository, exchanging both products and services.[3]

The decision facing a software developer is whether to build the program from scratch or to reuse existing software. From an economic point of view, the decision is clear: if the cost of acquiring and integrating a reusable component into the system being developed is less than the cost of developing the code from scratch, then it pays to reuse. Repositories serve to reduce the costs of both acquiring and instantiating reusable assets. A repository reduces the cost of acquiring assets by facilitating their search and retrieval. This is important because, to reuse a software artifact effectively, you must be able to find it and incorporate it faster than you could build it [8].

The way a repository serves to reduce the instantiation costs is a bit more subtle. The integration of an asset into a new context involves several costs, such as the costs of adaptation, modification, parameterization, and tailoring. For example, the reuse of a component that is being reused verbatim will usually still require effort to define its input, such as actual parameters or tables of input. To the extent that a software repository

---

2 Throughout the paper, we refer at times to "repositories," "the repository," and "a repository." Unless stated otherwise, these are all meant in a generic sense and should be interpreted to apply to any repository.

3 Although not direct contributors or subscribers, we must also include government acquisition offices in this discussion because they can bring to bear a great deal of leverage in fostering the reuse marketplace by awarding contracts based on the demonstrated reuse performance of a contractor and by requiring that designs and implementations conform to standardized domain architectures.

enforces a standard[4] that defines the parameters or tables, it serves to make generalization easier, less costly, and thus more likely to occur.

B. ENCOURAGING REUSE WHILE RECOVERING COSTS

As with most investments, the costs associated with reuse in general and the repositories in particular are incurred early in the process while the benefits accrue later. In an era of decreasing DoD budgets, investment dollars are dear, and managers are under considerable pressure to make investments pay back as soon as possible. Repositories are thus faced with two potentially competing objectives: to encourage reuse on the one hand and to recover costs on the other.

One way to manage repositories (and recover costs) is through a fee-for-service arrangement. Such an arrangement involves charging a fee to the subscribers to the repositories and using the proceeds to reimburse the repository for costs incurred in providing their products and services. The basic tenet of the fee-for-service arrangement is that it provides a means by which a repository can recover its costs and be self-funding.

A repository could encourage reuse by not charging for its products or services or by charging only the marginal cost of dissemination. Most likely, the repository would not recover its costs under this option. On the other hand, the repository could ensure it covered its costs by charging an average cost for every product or service. (To obtain an average cost, the repository could sum its costs and divide the total by the number of units it expects to sell.) However, this option would lower the quantity of reusable assets demanded. Indeed, if costs were too high, customers might not be willing to pay, thus suffocating any reuse. How can repositories set prices that strike a balance?

C. OBJECTIVE OF THIS STUDY AND PREVIOUS STUDIES

The objective of this study was to examine the options facing government software repositories under a fee-for-service structure and assess their economic impact. We were particularly interested in how best to encourage reuse and recover repository costs.

Previous work for the Center for Information Management (CIM) at the Defense Information Systems Agency (DISA) focused on the cost recovery side of the fee-for-service arrangement [9]. For this study, we were given more latitude to look at how alternative cost-recovery strategies are likely to affect the reuse market. Our perspective

---

[4] In the vernacular of software engineering such a standard is called a "domain architecture."

allowed us to relax several constraints imposed on the authors of Reference [9]. For example, in terms of repository services, the authors of that report assumed that each repository service should recover its costs. Although we assumed that full cost recovery is an objective over the long term, we noted reasons why it may not be in the DoD's overall interest to make the repository self-sufficient if the cost of self-sufficiency is to diminish the nascent reuse program.

In terms of assets, the authors of the DISA/CIM report assumed that the repository was responsible for determining prices and that the fee charged for an asset should be a function of its development cost (or the acquisition cost to the repository). We assumed that the fee charged would be a function, not of cost, but of the value to the consumer. In addition, rather than assuming that the repository is responsible for owning and pricing assets, we examined the implication of private ownership and pricing.

The ownership or property rights issue goes to the heart of our economic analysis. Economic behavior is determined to a large degree by incentives. We ask the question: What are the incentives to both the subscribers and contributors to behave in a manner that is beneficial to DoD? We believe that understanding the economic environment in which pricing decisions are made is an important first step towards making them effectively, and that property rights are important in determining that economic environment.

Another recent paper on the topic of fee for service [10] does an excellent job of laying out the case for intra-organization fee for service. The focus of that paper is automated data processing (ADP) services. However, it does not deal with several issues addressed in this study. Specifically, it does not cover the case where subsidies may be appropriate. Since its focus is ADP services, it does not address the unique characteristics of software and the nature and the costs incurred by a repository that make it difficult to follow conventional fee-for-service pricing schemes.

D. OUTLINE OF THIS PAPER

The next chapter presents a high-level framework for reuse and explores the role of repositories within that framework. It details the type of reusable assets that reside in the repositories and the ancillary services that facilitate the flow of software artifacts into and out of the repository. A summary of how several repositories currently operate is also included.

Chapter III presents a theoretical construct of a software repository. It explains how a repository serves as a "market intermediary," bringing subscribers and contributors

together. The construct demonstrates how a repository serves to lower the costs of developing software by reducing the transaction costs of DoD software developers and users. Chapter III addresses the question of the optimal price for repository assets and services and explains how the theoretical optimal price leads to a situation in which the repository could not recover its costs of providing assets.

Chapter IV discusses several issues, including property rights, that greatly influence pricing strategies within the DoD. A good deal of the chapter is devoted to the issue of government versus private property rights.

Chapter V explores candidate pricing schemes for repository services and assets.

Finally, Chapter VI summarizes the findings and presents several recommendations for pricing of repository assets and services.

II. GOVERNMENT SOFTWARE REPOSITORIES

The objective of this chapter is threefold:

- to present a high-level framework for reuse and describe how repositories fit into that framework,

- to outline the role of repositories in the framework in terms of their assets and services, and

- to summarize current repository practice, particularly with regard to the pricing of assets and services.

A. FRAMEWORK FOR SOFTWARE REUSE

Figure II-1 presents a framework for software reuse. This framework was developed as part of DoD's Software Technology for Adaptable, Reliable Systems (STARS) program. It represents a vision for effective reuse. Note the three distinct realms of activity. The first, asset creation, consists of the development of generalized assets according to a specific domain model and architecture. The activities in this realm are often referred to as "domain engineering." The second, asset management, consists of activities involved in cataloging, storing, and retrieving assets and is primarily the responsibility of a repository. The third area, asset usage, consists of integrating assets into a specific application program and is usually referred to as "application engineering." Keep in mind that the assets being created, managed, and used may consist not only of code but the entire realm of life-cycle products, including requirements, designs, test suites, and documentation.
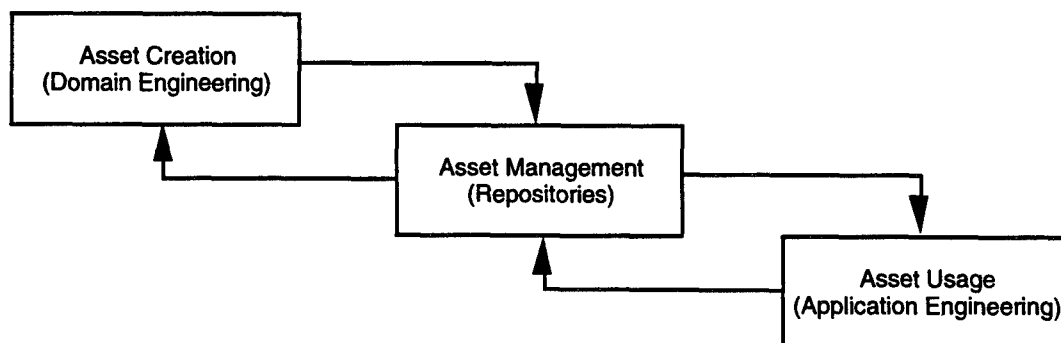
Figure II-1. Managed Reuse

Figure II-1 represents a vision of managed reuse. In contrast to that vision, much of the reuse that occurs today has been labeled "ad hoc" or "opportunistic" and takes place outside of the three distinct areas of activity. Ad hoc reuse consists primarily of code segments that must be modified in order to be used in a new application. The reuse that occurs most often takes the form of programmers using their own code and less often someone else's code on the same project [11]. Even less reuse occurs across projects, and still less across organizations. In the absence of asset management, "uncontrolled cloning" [12] can occur in which code segments are copied and modified. Uncontrolled cloning makes distribution of corrective, adoptive, or perfective maintenance upgrades impossible, because it is not known where copies of the component reside.

Effective reuse demands an asset management process. The specific activities underlying that process are discussed in Section B.2.

B. THE ROLE OF REPOSITORIES

We have chosen to segregate our discussion of repository products into two categories: assets and services. The reason for this distinction has to do with the different charging schemes these two types of products engender. The next section lists the assets provided by a repository. Section B.2 explores the issue of services.

1. Repository Assets

Repository assets consist of any software-related outputs, including the following:

- **Source Code:** The program instructions processed into machine code by some combination of preprocessors, compilers, and assemblers.

- **Executable Code:** The output of compilers and assemblers that is executed by the computer hardware; machine code.

- **Domain Models:** A definition of the functions, objects, data, and relationships in a domain, consisting of a concise representation of the commonalities and differences of the problems of the domain and their solutions. A domain is a set of applications that perform common sets of functions. Examples of DoD software domains are avionics, command, control, communications, and intelligence (C3I), and logistics.

- **Domain Architectures:** A high-level description of the objects and functions within a domain and their interfaces and data flows.

- **Requirements:** A description of the functional behavior, performance constraints, and external interfaces for a software system.

- **Preliminary Designs:** A description of software components, their interfaces, and data flows.

- **Detailed Designs:** A description of the internal logic of software components.

- **User Manuals:** Tutorial and other material to assist the user in accessing system functions.

- **Test-related Products:** Material consisting of test plans, test procedures, and test cases used in exercising or evaluating systems or system components by manual or automated means to verify that they satisfy specified requirements or to identify differences between expected and actual results.

We can distinguish among three categories of software assets based on the degree to which they were designed for reuse. One category consists of existing software assets developed without reuse in mind. They are likely to require a fair amount of modification to be reused in another application. A second category consists of existing assets also developed without reuse in mind but that have been improved and generalized. The third category consists of assets designed from the beginning to be general and reusable.

We discuss each of these categories later in this paper (Chapter V). The important point from an economic perspective is that creating assets in the second and third categories require investment. Some person or organization has to add resources to enhance the reuse potential of any given asset. Although these two categories require up-front investment, they also yield the greatest benefits.

2. Services

Software repositories function much like a public library. They have a means for classifying and cataloging assets, much like the Dewey Decimal System for books. They allow users to browse for and checkout (download) assets. Such services represent the minimum that a repository might provide. Additional services include consulting and education and assurance that the assets meet certain quality standards. In addition, assets must be maintained and their configurations must be controlled. It is especially important to keep track of which users have which versions of which assets. While several of these services could be performed within or outside of a repository, they are all part of asset management.

The potential repository services are:

- **Cataloging Assets.** This service includes accepting an asset into the repository and classifying it according to a cataloging structure. The cataloging structure and classification scheme must be in place before the repository can accept assets. The sophistication of cataloging schemes can vary drastically. They can be as simple as keywords or as sophisticated as a domain architecture. Once the classification scheme is in place, each asset must be classified according to that scheme and entered into the repository.

- **Certification.** A form of quality assurance, certification focuses on the non-functional aspects of an asset. Several repositories (e.g., the Asset Source for Software Engineering Technology and the Defense Software Repository System) follow a multi-level certification scheme ranging from visually checking that all files and documentation are included to compiling and executing code and running it through source code analyzers.

- **Qualification.** This focuses on the functional and performance aspects of an asset and is done in the context of a domain model and architecture. If done well, qualification will remove a substantial amount of risk for subscribers to the repository because they will have some assurance of what they are getting. They will know, for example, that an asset functionally does what it is supposed to do and that it conforms to all interface specifications. In addition, its performance characteristics could be delineated.

- **Searching for Assets.** Providing subscribers with convenient and efficient search mechanisms is a key repository service. In our view, browsing through the repository is an activity that should be encouraged. The more subscribers browse, the more likely they are to find and utilize appropriate assets. Searching could be done on-line or off-line. Allowing subscribers to download portions of the catalog so that they can browse off-line will free up resources. Once domain architectures are in place, specifying and searching for a needed asset should be much easier and might involve, for example, simply pointing to the desired component on a graphic representation of the architecture.

- **Maintaining Assets.** This service includes defect correction as well as performance and functional upgrades. In our view, asset maintenance should not be the responsibility of the repository but of the asset developer. Regardless of who is responsible, maintaining assets will be a necessary service that could be coordinated through the repository.

- **Managing Asset Configuration.** As assets are upgraded, version identification procedures will be needed to keep track of which subscribers have which versions. Such information is especially important to know when problems are reported. The maintenance organization needs to know which version has the problem. Once corrections are made, it is important to know which users need the corrected version.

- **Enforcing Data Rights.** Different assets may be associated with different data rights. Some assets may be in the public domain and available to anyone. Others may be available only to government employees and contractors. These data rights must be enforced by making sure that the appropriate people have access to the appropriate assets.

- **Reporting Problems.** This service provides subscribers with a mechanism for reporting problems encountered while using the assets. It also provides for the successful resolution of problems through either a correction to the asset (if the problem lies there) or help to the user (if the problem lies there). Problem reporting could be done within the private sector or by the repository.

- **Disseminating Assets.** This category refers to the physical distribution of assets which could be electronic, on a diskette, or even on paper.

- **Basic Subscriber Services.** This category includes any of the basic services that a subscriber to an electronic database would expect. These include setting up accounts, providing a help line, and providing access to bulletin boards.

- **Consulting, Educating, Training.** This category includes education and training for (1) asset developers on how to design assets that are reusable, (2) application developers on how to implement processes that incorporate reuse, and (3) acquisition personnel on writing requests for proposals and contracts that incorporate and reward reuse. This service could be done within the private sector or by the repository.

C. CURRENT PRACTICE

We surveyed several existing repositories to find out about current practices. We were especially interested in the following questions:

- How does the repository obtain assets?

- Does the repository charge subscribers for assets or services?

- If so, who sets the fees and on what basis? (Is it the repository or the contributor?)

- Does the repository keep all fees or are they shared between the repository and the contributor?

We spoke to representatives from three existing software repositories: DISA/CIM's Defense Software Repository System (DSRS), the STARS program's Asset Source for Software Engineering Technology (ASSET), and NASA's COSMIC. We summarize the findings from those conversations here. More detailed information about each repository is included in Appendix A.

II-5

Possible alternatives for obtaining assets for a repository include coercing potential contributors via government directives, paying for assets through licenses or purchasing, and seeking out assets in the public domain. We found all three of these alternatives being used in practice. The assets in COSMIC are obtained via a NASA-wide directive requiring that components developed by NASA employees or contractors be submitted to the repository. The components in ASSET are all in the public domain; most were developed under sponsorship of the STARS Program. DSRS includes assets from commercial vendors, such as the Booch components, obtained by negotiating DoD-wide licenses; the majority of the assets were developed under DoD funding.

Of the three repositories, only NASA's COSMIC charges for assets (although assets are provided without charge to NASA employees and NASA contractors). The fees, set by COSMIC, are based on factors including the type and number of hardware platforms on which an asset runs and the number of lines of source code. The contributors to COSMIC receive a modest award of several hundred dollars when an asset is accepted into the repository. All money collected as fees goes to the repository to cover expenses. While this cost-recovery mechanism has been reasonably successful, one area for improvement would be the addition of incentives for the contributors to contribute and maintain high-quality assets. Currently, the quality of assets submitted varies and no strong incentives exist for their maintenance.

ASSET charges nothing at present, although the plan is for the repository to be fully self-supporting within five years. ASSET views services rather than assets as the more likely source of funds. DSRS plans to change to a cost-recovery operation within the next few years as well.

III. THE ECONOMIC ROLE OF SOFTWARE REPOSITORIES

This chapter describes the economic role of software repositories in bringing contributors and subscribers together. The ultimate goal of the repository is to lower the cost to DoD of developing and maintaining software. The argument presented in this chapter focuses on how the repository serves to achieve this goal. Specifically, a case is made that the repository reduces the risk and uncertainty of both the contributor and subscriber of software products and services. The reduction of risk and uncertainty has the effect of reducing the costs to the contributors and subscribers. Savings are ultimately passed on to the DoD.

A. THE REPOSITORY AS A MARKET INTERMEDIARY

Providing a product or service consists of several steps: (1) development or purchase of input (e.g., raw materials), (2) design and production, (3) advertisement and sales, and (4) distribution. In some cases, one organization performs all of the steps itself. In other cases, the organization performs some of the steps itself and makes arrangements for other steps to be carried out by other parties.

A primary consideration in deciding upon the optimal form of the organizational structure is transaction costs. Transaction costs are those costs not directly incurred in the physical process of production. They include costs such as gathering information; negotiating, drawing up, and enforcing contracts; delineating and policing property rights; monitoring performance; and changing institutional arrangements. All other things being equal, an organization would be willing to subcontract out some task if the cost of the product or service plus the cost of negotiating and monitoring the arrangement is less than the cost of producing the product or service in-house.

One form of economic organization that has evolved as a means to reduce transaction costs is the market intermediary. A market intermediary serves to bring buyers and sellers together. Two excellent examples are the equity and bond markets in the United States. Other examples are real estate brokers and retail stores.

Software repositories are another example of a market intermediary. The repository acts as a liaison between contributors of software products and services and subscribers to those products and services. Organizations that serve as intermediaries usually perform two

main functions: they lower the cost of exchange and they provide information that reduces the uncertainty about the level of market prices. Lowering costs and reducing uncertainty tends to result in lower prices and additional quantities being sold. Also, because intermediaries lower the cost of exchange between different organizations, the existence of market intermediaries sometimes permits the organizations to specialize in one step or process of production when it is effective to do so. This also results in lower costs.

The separate effects that a repository has on the contributors and subscribers to the repository are presented in the next section.

B. HOW A REPOSITORY LOWERS COSTS FOR CONTRIBUTORS AND SUBSCRIBERS

The repository's role as market intermediary and how that role affects the contributors to the repository can be more formally seen in a series of diagrams presented in Figures III-1 and III-2. In these figures, each graph has two axes labeled P and Q. P is denominated in dollars and shows the cost of different quantities (Q).[1] Panel A of Figure III-1 depicts the cost structure for an organization (Organization 1) in a new industry typified by software development. The organization performs an agglomeration of processes, such as purchasing input, transforming input to output, and marketing final products. For simplicity, assume the organization performs two processes, process A (development of application code) and process B (development of encryption code). These two processes are described by average cost curves AC_A and AC_B, which sum to the organization's average total costs AC_F. The average cost curve represents both the fixed costs and the variable costs incurred by the organization for any given level of output. The curve tends to fall as output increases because the fixed cost are spread over a greater number of assets until a point is reached at which additional costs must be incurred (overtime, more powerful computers, more storage, etc.) and the curve tends to rise as more software is developed. In a new industry such as software development, the organization often has no choice but to manage processes A and B internally, because it is virtually the only organization in the new industry.

As the industry grows, however, additional firms enter and the industry-wide volume of both process A and process B increases. Eventually, industry volume is large enough to support an organization (Organization 2) that specializes in writing encryption

---

[1] For an interesting application of transaction costs in a different industry, see Barkema and Cook [13].
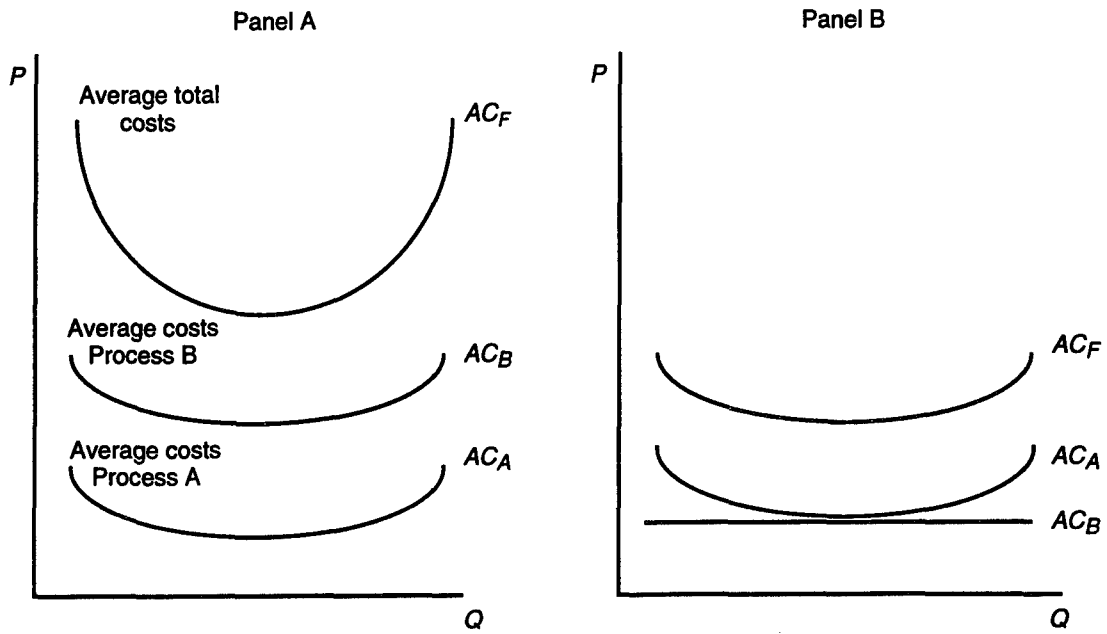
Figure III-1. How Specialization in Software Development Lowers Average Total Costs
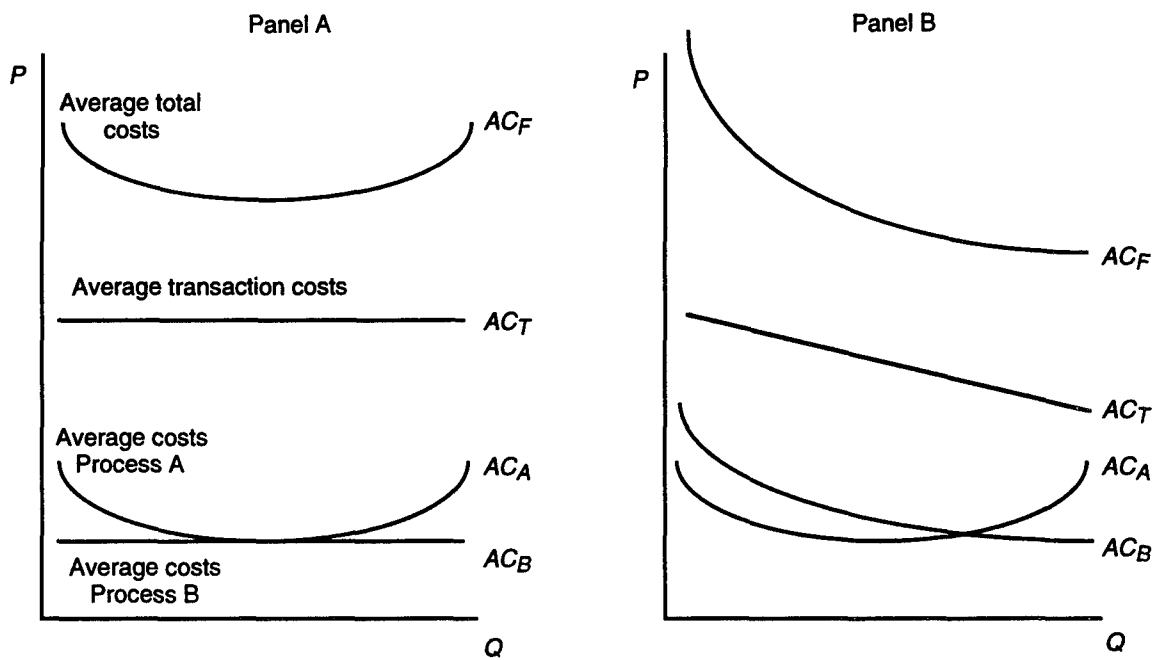
Figure III-2. How a Repository Lowers Transaction Cost

III-3

algorithms (process B), exploiting economies of scale unavailable to the original pioneering organization.[2] Now, Organization 1 can lower its total development costs by relying on Organization 2 for process B, which it can buy at a price lower than its own development costs regardless of volume (Panel B, Figure III-1). This simple example suggests that horizontal production (in which all of the resources needed for development of a particular application are not owned by a single firm) should become more common and vertical integration (in which all of the resources of production needed for the development of a particular application are owned by a single firm), less common as a new industry such as software development grows.[3]

The discussion so far has focused on the contributor's production or development costs and ignored transaction costs, the costs of managing marketing relationships, which can change the picture markedly.

Panel A of Figure III-2 again shows the cost structure of the original pioneering organization (Organization 1) after its decision to rely on Organization 2 for process B. But Figure III-2 also accounts from Organization 1's transaction costs (AC_T)—the costs of managing its relationship with Organization 2. The transaction costs might reflect the cost of searching for a supply of encryption code. The addition of these costs significantly pushes up Organization A's total average costs (AC_F).

As an alternative to using the traditional marketplace in which Organization A searches for subcontractors who will provide the encryption code, the two organizations may be joined through the repository. The repository works to influence the transaction costs (AC_T).

The repository also lowers the cost of exchange for subscribers by providing a focal point for gathering information about products, services, and prices—one-stop shopping. The repository also has the effect of reducing the uncertainty about market prices for both the contributors and the subscribers by providing this focal point that allows potential participants to more easily understand the approximate conditions under which they will have to buy or sell their products or services. In the absence of a repository,

---

[2] One source of economies of scale in writing encryption software (process B) is mastering the complex rules for security algorithms. Once an organization invests the time and effort in understanding the regulations and protocols detailed by the National Security Agency, it can then apply the lessons learned to future products more easily.

[3] There are special cases where, as the market grows, organizations specialize and vertical integration becomes less common. That is, as organizations grow, they tend to rely less on outside contractors. Legal services come to mind as an example.

market uncertainties could prevent both the contributors and the subscribers from making large fixed investments in their businesses.

Lower marketing risks could embolden contributors to make large investments in code development (e.g., development tools), driving down development costs by capturing economies of scale and economies of scope and thereby reducing total costs AC_F. This scenario is depicted in Panel B of Figure III-2.

C. THE BENEFITS OF A REPOSITORY TO DOD

The previous section focused on the repository's market intermediary role and how that role can lower the costs to both the contributor and subscriber of the repository. What is the benefit to the DoD of these lower costs?

To answer that question, we must introduce the concept of the demand curve and its relationship to the cost curves illustrated in Figures III-1 and III-2. The curve labeled "Demand" in Figure III-3 represents the amount (Q) of a given product or service the subscriber (DoD) would purchase at various prices (P), all other things held constant.
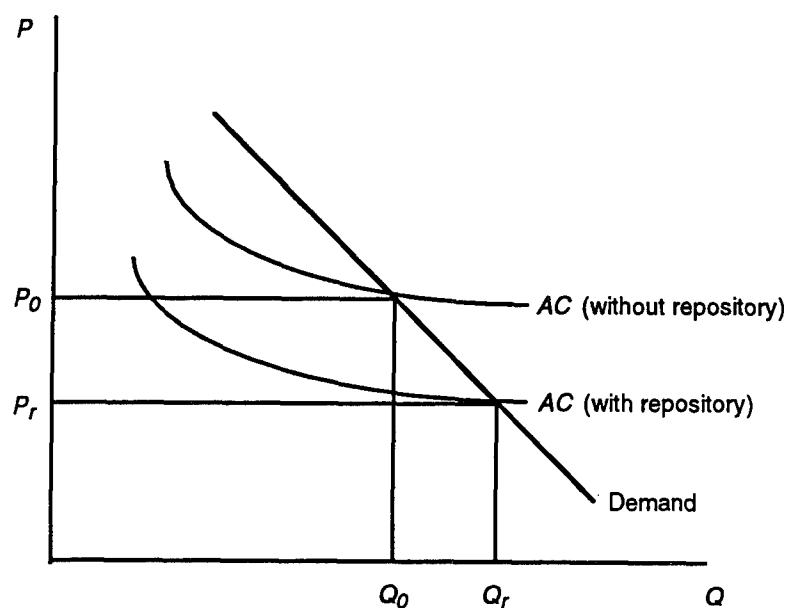
Figure III-3. Repository Benefits

The downward direction of the curve indicates that more of the product or service will be demanded at lower prices than at higher prices. If the organization were to charge any price lower than the average cost (i.e., any price that lies below the curve AC) it would result in a loss to the organization. Without the repository, the organization would face the top-most average cost curve. The organization would need to charge *at least* price P_0 for its

products and it would produce Q_0 quantity of the product. However, with a repository's beneficial effects of reducing transaction costs, thus lowering the average cost curve (as depicted by the lower cost curve), the organization can now cover its costs by charging only P_r and offering a greater quantity of the product at the lower price. In a competitive market, these cost savings will be passed along to the government via lower bids by government contractors.

To review, we have argued that the economic role that repositories play is one of a market intermediary. In this role, the repository reduces the risk and uncertainty of both the contributor and subscriber of software products and services. The reduction of risk and uncertainty has the effect of reducing the costs of the contributors and subscribers. It also has the effect of allowing the economic actors to specialize and capture economies of scale in their operations. Economies of scale also have the beneficial effect of lowering unit costs. The government benefits by way of the fact that those contributors and subscribers can charge lower prices for their products.

There is, however, a cost to be paid for these benefits: the cost of establishing and operating the repository. The following section addresses the question of what economic theory tells us about how to set prices for repository products and services so that we strike an optimal balance between encouraging reuse and recovering costs.

D. DEMAND AND COST CURVES

Setting the price for repository products or services involves examining the demand and cost curves faced by the repository. Just as the contributors and subscribers of the repository have demand and cost curves, the repository also faces its own curves. Figure IV-1 presents the repository's total demand curve for reusable assets.[4] Price is measured on the Y axis and quantity, on the X axis. In this case, price represents the price that the repository charges the subscriber. For now, it does not matter whether the asset is a piece of code or is some repository service such as search and retrieval. The quantity is some measure of the number of things (code or services) demanded. The demand curve

---

[4] Markets exist only if potential contributors believe they can generate enough sales to at least cover their costs and potential subscribers believe that goods (in this case reusable software assets) will be available for sale at prices that will make the goods cost-effective to purchase. This means that there must be some quantities where the projected price that subscribers are willing to pay (shown by the demand curve) is equal to or higher than the projected costs of the contributor's production (shown by the average cost curve). Once there is at least one such point, a market is possible; once there is more than one such point, we can try to decide which point is better by some criteria. We assume that such points exist.

represents the amount the repository subscribers would purchase at various prices, all other things held constant. As before, the downward direction of the curve indicates that more assets will be demanded at lower prices than at higher prices. If the only goal of the repository were to promote reuse, it would obviously set the price at zero and give the assets away. But because the repository must pay for its operations, the price must be greater than zero, or the repository must be subsidized.

The constraint of paying for the repository's operations is depicted by the average cost curve AC. The average cost curve is related to the marginal cost curve (MC). The marginal cost curve measures the increment to total costs per unit addition of reusable assets.
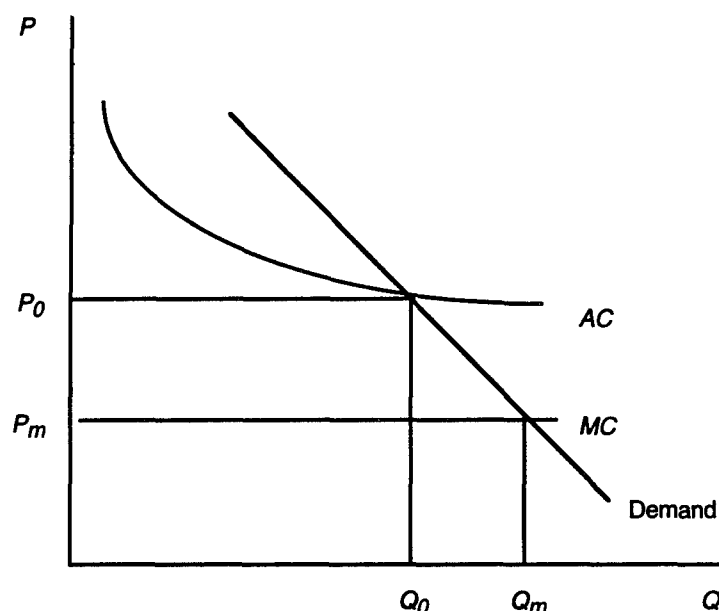
Figure III-4. Repository Products Demand and Cost Curves

We believe that the demand and cost curves for repository assets would be different than those for repository services. We hypothesized repository assets exhibit cost curves similar to those shown in Figure III-4. Curves of this nature are typical of the U.S. pharmaceutical industry. The large fixed costs of bringing a drug to market (from research and development and from regulatory requirements) often dwarf the marginal cost of actually producing the next additional capsule or pill. Thus, the marginal costs are well below the average costs and are essentially constant over the range of output. The average costs are much higher than the marginal costs where there are few units sold.

Note that the magnitude of the fixed costs (and thus the shape of the cost curves) depends to some extent on the mechanism used to bring assets into the repository. For example, under a repository scenario in which the repository purchased the assets, the fixed costs would be higher than under a scenario in which assets were given to the repository as a result of a government directive, were purchased by another government organization, or were placed under consignment by the asset developers.

On the other hand, services provided by the repository (e.g., consulting and training) are hypothesized to exhibit cost curves like those depicted in Figure III-5. Fixed costs are small compared to the variable costs (i.e., cost of the analysts' time).
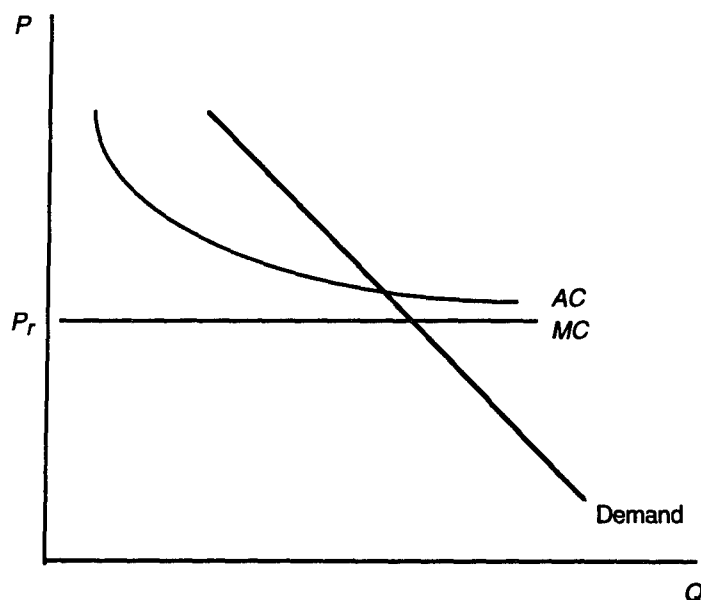
Figure III-5. Repository Services Demand and Cost Curves

In the cases depicted above, the average cost would rapidly, asymptotically approach the marginal cost curve as the number of assets produced by the repository increased. Cost curves of this nature have implications for the price a repository can charge and still recover costs.

If the repository were to charge any price lower than the average cost (i.e., any price that lies below the curve AC), it would result in loss to the repository. Any price above the average cost curve would result in a "profit" to the repository.

So where should the repository set the price? Economic efficiency requires the price be set equal to the marginal costs (i.e., the cost to the repository to provide the last unit of reusable asset). The reason that price equaling marginal cost is considered efficient is that price is a measure of the value of the last item to a purchaser, and marginal cost is the cost to produce the item for the supplier. If fewer units are purchased, the value to purchasers as measured by the price that purchasers are willing to pay for the forgone units is greater than the cost of producing them. If more units are purchased at the price purchasers are willing to pay, then the value of these units is less than the cost of producing them.

E. PROBLEMS WITH THE PRICE-SETTING RULE

Implementing the rule of setting price equal to marginal cost poses several problems: (1) for many assets and services, setting price equal to marginal cost will not cover all costs, (2) sometimes the costs and demands in the market do not reflect all the costs and demands for the asset or service, and 3) some of the characteristics of software that are lumped under the heading of "intellectual property" generate difficulties in creating a functioning market for reusable components. These problems are explored in the following subsections.

1. Prices May Not Cover Costs

In situations where the fixed costs are large relative to the marginal costs, the repository will not recover its costs under the price-equals-marginal cost rule. We believe large fixed costs and small marginal costs are typical of repository assets. The repository would always lose money if it charged the marginal cost since the marginal cost is always less than the average cost of producing the asset as shown by the average cost curve. In such a scenario, the repository would need a subsidy. This subsidy could be paid up front by subsidizing the fixed costs, hence making the average cost curve closer to or the same as the marginal cost curve, or as a per-unit subsidy.[5]

---

[5] A subsidy raises a new issue. The government must raise revenues (generally through taxation) to make up the deficit between marginal and average costs. Are the benefits to society derived from the subsidy larger than the cost imposed on society by the tax? Answering this question involves a maze of "second best" calculations. Unfortunately, obtaining accurate data for these calculations from a market that does not yet exist is probably impossible, and certainly beyond the scope of this task.

Another way to mitigate the problem of not covering costs is through a different arrangement of ownership of the assets. This solution is discussed in detail in Chapter IV.[6]

There is also a practical problem with setting the price equal to the marginal costs. As previously demonstrated, above, economic theory dictates that the repository set the price of its assets and services equal to the repository's marginal cost. In practice, obtaining marginal cost figures is often more difficult than acquiring average cost data due to several factors. Almost all cost accounting information is in the form of average or total rather than marginal data. By its very nature, marginal information often represents the answers to hypothetical questions beyond the range of the organization's actual experience. For example, what will be the effect on the repository's profits of an increase in expenditures on new technologies that allow for easier browsing, whether or not the repository has ever tried it?

When marginal data are difficult or impossible to come by, it is necessary to make do with average figures. Rough adjustments to the average data can be made to bring them closer to the unknown marginal figures.[7]

---

[6] Economists have worked out another solution (besides providing a subsidy) to the problem of pricing when there is a divergence between the average and marginal costs. This solution is sometimes referred to as the Ramsey pricing problem [14]. Simply stated, the Ramsey solution states that the enterprise should price its products in such a way as to charge more for products that are inelastically demanded. Elasticity measures the percentage change in quantity demanded in response to percentage changes in the price of the good. When there is a weak percentage response (say, the price goes up by 10 percent but the quantity demand falls by only 1 percent), then demand would be classified as inelastic. If, on the other hand, the price goes up by 10 percent and quantity demanded falls by 15 percent, then demand would be classified as elastic. The reasoning behind the Ramsey solution is that the allocation of inelastically demanded goods are least affected by the price increase. The principle has been widely recognized and accepted by economists and practitioners. For example, since 1983, the Interstate Commerce Commission adopted Ramsey pricing as the underlying principle it would follow in the regulation of railroad rates. However, it is not clear that DoD procurement regulations would allow such price discrimination among products as called for by the Ramsey solution. Also, detailed information as to the elasticity of various repository products would be needed in order to implement such a pricing solution. This is beyond the scope of the present effort.

[7] This is especially true of the type of cost data available from repositories and the situation depicted concerning the market for reusable assets and services. If there is reason to believe there are economies of scale (falling average costs), marginal costs will be less than average costs, so the average cost figure must be adjusted downward to yield a better approximation to the marginal cost figure. We believe reusable assets do exhibit economies of scale. Such economies of scale would indicate that it will be difficult to make use of the repository's average cost data as approximations to marginal cost data. Some analysis of the repository's cost data would be required in order to compute the exact adjustment needed to align marginal and average costs. However, for repository services we believe there is little difference between the average and marginal costs. This lack of difference would suggest that average cost data could be substituted for marginal cost data when the optimal charge for repository services is being analyzed.

2. Market May Be Too Small

Another problem with setting the price equal to marginal costs has to do with the size of the potential market for reusable assets. It is likely that many assets in the repository will be applicable to vertical domains such as avionics or command and control.[8] In most cases, such assets will not have a potential market in the hundreds or thousands. It may be five or ten. In such a case, the high fixed costs can be amortized over only a few users. This will make the discrepancy between what should be charged in order to promote cost-effective reuse (i.e., price equal to the marginal costs) and what is necessary to cover costs (i.e., average costs) larger at low demand than at higher levels of demand.

3. Intellectual Property and Public Goods

Software has characteristics or attributes that can make developing a market in reusable software components difficult. Such characteristics are often lumped together under the heading "intellectual property," in large part because of the laws that have sprung up to protect them.[9] Non-rivalness is the underlying characteristic that creates the problems intellectual property law attempts to correct. Non-rivalness means that one individual's use of software does not preclude another from using it. Products and services that have these characteristics are often referred to as public goods.[10, 11] For software, the non-rivalness occurs because almost all costs of producing a particular software asset accrue from the production of the first unit, and additional units are essentially free. Because costs accrue in this pattern, the divergence between the average cost and the marginal cost of existing software is large.

Intellectual property law attempts to do the same thing for all intellectual property that we are attempting to do for software: create incentives that balance the cost to purchasers and the revenue to suppliers. Intellectual property law usually does this by

---

[8] A vertical application domain is defined to be the knowledge and concepts that pertain to a particular computer application. A horizontal domain is defined to be the knowledge and concepts that pertain to a particular functionality of a set of software components that can be used for more than one application domain (e.g., user interface packages). These definitions were taken from [15].

[9] For an introduction to this topic, see Besen and Raskind [16].

[10] Another characteristic that is sometimes associated with public goods is non-excludability, which means it is difficult for people to exclude others from using the good (in this case, software). Software can be constructed so that people can be excluded from using it by techniques such as copy protection, but the market has rarely supported this approach except in the case of computer games. Better solutions involve giving individuals incentives to pay for the software. This can be done through legal mechanisms (raising the costs of not paying), or charging for other aspects of software such as documentation, upgrades, and technical support. This approach is often used for shareware.

[11] For a more detailed exposition on the reasons this character led to market failure, see Bator [17].

creating property rights. Therefore, a major portion of the next chapter is devoted to our analysis of what property rights should be given to various participants in the repository marketplace in order to offset, or even use to advantage, the public good characteristics of software.

IV. PROPERTY RIGHTS

The previous chapter presented a theoretical discussion of the optimum price for repository products and noted the problems with following the pricing rule. In this chapter, we address an issue fundamental to price setting—to whom should the property rights of the repository assets be assigned? That is, who decides how much assets will cost and how they will be used. We argue that, in the main, private organizations should maintain significant property rights to software assets and that the government should acquire property rights only in specific, limited areas.

A. GOVERNMENT PROPERTY RIGHTS

Economists define property rights as the exclusive authority to determine how a resource is used and the right to delegate, rent, or sell any portion of it, whether owned by the government or by an individual. Up to this point, the assumption has been that the repository will control the property rights to all the assets that actually reside in the repository. As noted in Chapter II, this is the most common DoD practice.

A case can certainly be made for the repository to own the property rights of certain types of assets. For example, for reasons of national security, the government must own or at least control assets that are classified. In addition to national security assets, ownership and control of domain models and possibly domain architectures might also be beneficial to the DoD [18]. Domain models should be under government ownership and control because of their critical role as standards for developers of components and for developers of application systems using those components.

Domain models, in effect, set the ground rules for a domain and define the basic entities and their relationships. Domain architectures lay out the high-level components and their interfaces. For some domains, the government could control both the domain model and a single standard architecture. For other domains, it might be preferable to allow competing architectures.

There are negative consequences of the government repository owning and controlling other types of assets. Ownership implies responsibility and responsibility carries with it costs. If the repository is the owner of the assets, it assumes responsibility

for four areas: predicting market demand, pricing assets, enforcing data rights, and supporting and maintaining software.

The first area is in predicting market demand in order to know which components are worth investing in. The objective is to invest in assets for which there is a demand and to avoid investment in assets for which demand is low.

A second area has to do with pricing assets. Setting prices and observing their effect on demand is done well by the private sector. We return to this area of responsibility in the next section.

A third area has to do with enforcing data rights. The data rights associated with an asset determine who is allowed access (for example, government employees and contractors versus the general public). Under government ownership, the government is responsible for ensuring that data rights are enforced. Data rights enforcement has been a problem already for one repository (DSRS), which has been under pressure to put assets into the public domain. DSRS personnel have had to search the history of each asset to determine specific data rights. This task has proved to be difficult and time-consuming, so only a small percentage of DSRS assets have been placed in the public domain.

A fourth area has to do with support and maintenance. If the government owns the assets, it is responsible for their maintenance. Even if it contracts out for the service, the government's taking on this responsibility will substantially expand the resource requirements of the repository. A report by Softech for DISA/CIM [19] made a strong case that, for reasons of incentives and history, the original developers of a component are almost always in the best position to provide maintenance and support.

B. GOVERNMENT PROPERTY RIGHTS AND INCENTIVES TO REUSE

Property rights also have implications for the economic incentives of both contributors and subscribers. Software is made reusable by making it generalizable. Generalization involves removing some of the problem-specifics from a software component and requiring the user of the component to re-supply those details as parameters that describe the specific context of reuse. Appendix B provides an example of generalization.

There is a cost to generalizing code (and thus making it more reusable). A detailed study of generalized software development at the Flight Dynamics Division of NASA's Goddard Space Flight Center found that the cost of developing completely general software was nearly three times the cost of developing single-purpose code [2].

In the NASA process, a set of reusable software components was developed over time by repeatedly generalizing the solutions to previous systems that were in the same domain. This perspective over time and across several sets of related requirements allowed the developers to derive a set of reusable components that reduced the need for software development for a new system in the domain by as much as 90 percent.

The key to the success of this effort was the generalization of a pre-existing solution rather than the adaptation of the solution to the new context. The software was generalized so that it would satisfy both old and new requirements. Over time, this practice caused the software components to be increasingly more general until they could satisfy new sets of requirements in the same domain. The effort to generalize and apply a pre-existing software solution so that the resulting generalization would satisfy both old and new requirements was even more costly than writing a new solution from scratch. Once generalized, however, the benefits for future systems were substantial.

These findings support findings reported in the research literature indicating that, to make reuse worthwhile to potential subscribers, the existing code needs to be generalized. The NASA study also illustrates the significant additional cost (i.e., 200 percent) incurred in generalizing existing code.

From an economic standpoint, the interesting question is: "Who has the incentive to incur these additional costs under the regime of repository ownership?" One would suspect that government sponsors who pay for software development (e.g., weapon system program mangers) would be reluctant to double or triple their programs' costs in order to make their code reusable by some other organization. Likewise, software developers would be reluctant to generalize their code if they could recover the additional costs either through royalties or cost recovery from the original government sponsor.

Short of coercion (i.e., mandates manifested in DoD instructions or contractual arrangements to write generalized code), there would seem to be little incentive to supply reusable code to the repository under the current ownership regime. Even coercion would seem to hold little real power since a small army of analysts would be needed to delineate what constitutes generalization and to subsequently enforce those rules on software developers/contributors.

C. PRIVATE PROPERTY RIGHTS

Private ownership of assets also implies some economic advantages and disadvantages. Under this scenario, the repository serves as a broker between contributors

and subscribers, charging a commission whenever an asset is checked out. Under private investment and ownership, predicting market demand, setting prices, enforcing data rights and supporting and maintaining software are worked out by the contributors to the market, not to the repository. With private investment, the risk involved in investing in assets will be spread across a broad base rather than falling on the repository.

Under private ownership, price setting would be done by the asset owners rather than by the repository. By allowing the market forces of supply and demand to come into play, pricing will be self-regulating: if a price is too high, the asset will not be in demand and the price should come down. Under private ownership, the asset owner knows the data rights associated with an asset and is responsible for enforcing those rights. Finally, maintenance and support become the responsibility of the owners and not of the repository. One of the primary advantages of private investment and ownership is that it will encourage product improvement. In most cases, the owners will be the developers; they know their product best and are in the best position to support and maintain it.

One economic aspect of private ownership argues against this arrangement. Under the private ownership regime, the owner will likely behave as a monopolist. A monopolist will tend to charge more for the assets than would the government. This arises because monopolists face no cost constraint—they already own the asset and will attempt to maximize their revenue without regard to cost because there is no cost. Thus, DoD may be worse off in the sense that subscribers could pay more under the private ownership regime than under a regime of government ownership. However, we believe that the positive effect of private property rights will outweigh this negative effect.

D. PRIVATE PROPERTY RIGHTS AND INCENTIVES TO REUSE

Private ownership provides a mechanism for software developers to recover the costs incurred in making the asset reusable. Software developers could be induced into generalizing their code by recovering these *additional costs* either through royalties or cost recovery from the original government sponsor. The problem under the regime of private property rights is not one of incentives. Rather, the problem stems from the nature of how software is treated in the world of DoD procurement.

Software is classified as intellectual property and is consequently covered by patent and copyright law. Typically, when the government enters into a contract for software, it is automatically granted a minimum set of *restricted rights* plus whatever additional rights are negotiated as part of the contract. Restricted rights include the right to execute the software on the computer for which it was acquired, to make backups, and to modify the software.

Ownership remains with the developing contractor who can then invest additional resources to "productize" (i.e., generalize and bulletproof) the component. This suggests the developer could make the code reusable and be free to charge the government again for the new productized component. However, one could argue that under the current government-restricted rights regime, the government has already paid for the code once. Therefore, the argument goes, why should the government be charged again for code that admittedly performs the same function for which the developer was originally reimbursed? How much does an asset have to change before it becomes productized?

Currently, these matters are decided on a case-by-case basis. However, case by case implies uncertainty. As we demonstrated in Chapter III, uncertainty raises cost, which, in turn, reduces the amount of reusable assets. Removing uncertainty reduces costs and induces more reuse. Resolution of this issue rests in the legal and institutional regulations arena [20]. However, there are pricing policies by the repository that can ameliorate this uncertainty. These pricing policies are presented in the next chapter.

E. GOVERNMENT-WIDE LICENSE: A SPECIAL CASE OF PRIVATE PROPERTY RIGHTS

A special case of private property rights involves government-wide licenses for privately owned assets. A repository could negotiate with contributors for favorable rates and/or pay the license fees to cover all DoD users. The latter practice is already being done by DSRS for limited sets of widely used assets (e.g., the Booch components).

Government-wide licenses combine the advantages of private ownership with large potential savings to the DoD. The case in which the license fees are paid by the repository is a clear form of direct subsidy. A subsidy can be applied to partially or completely offset the costs of specific assets and services in order to overcome transactions/information costs problems or to decrease the divergence between average and marginal costs. Also, a subsidy can be constructed to have different effects over time, either by arranging to subsidize a particular activity for only a particular period of time, or by subsidizing costs that tend to occur early in the process of creating a repository and the market for reusable components.

should be mandated and they should be provided at no charge. In all likelihood, domain models and architectures will not be developed and maintained by the repository but by other DoD organizations, each responsible for specific domains.

2. Repository Services

As shown in Table V-1, fewer services are the responsibility of the repository under the scheme of private ownership of assets. They consist of basic subscriber services (such as setting up computer accounts and providing a help line), cataloging, searching for assets, qualifying assets, and general reuse training and consulting. The pricing considerations for each of these services are discussed below.

a. Basic Subscriber Services

Account set up is a one-time cost per subscriber and providing a help line is a continuing service. The average and marginal costs are similar.

The average cost should be paid by the subscribers in the form of a yearly subscription fee.

b. Cataloging

The underlying cataloging taxonomy is based on the domain architecture of the domain in question. We believe that the cost of cataloging alone will be small (as opposed to domain analysis, certification, and qualification). The average and marginal costs are similar. The average cost should be paid by the contributors when an asset is accepted into the repository.

c. Searching

We believe the long-term goal is for all search costs to be paid by subscribers in the form of a commission when an asset is selected for use. However, we recommend that these costs not be charged during the search itself. Also, we recommend that the amount collected reflect the average search costs, not the costs associated with specific searches.

Standard commercial practice involves some way of displaying products for examination and comparison: potential purchasers browse merchandise in person or by catalog, and the purchase price includes some allocation of the costs of providing the salespeople and the space in a catalog.

d. Qualification

As the term is used here, "qualification" refers to a domain-specific activity. The objective of qualification is to ensure that a component's functionality conforms to its specifications and that its external interfaces conform to its domain architecture. As mentioned previously, we recommend that DoD policy mandate the use of domain models and domain architectures for mature domains. This will guarantee a market to potential vendors of components within those domains. We also recommend that only formally qualified components be listed within the repository and advertised as belonging to that domain. This will provide some assurance to potential users that the components will work as advertised and can be integrated into their applications.

It may not be the repository, but some governmental (or independent) entity will be responsible for qualifying assets, like Underwriters Labs does for electrical appliances. We discuss qualification here, however, because it plays an important role in encouraging a reuse market.

The fixed costs involved in developing qualification tests for any given domain will be substantial. The marginal costs involved in qualifying individual assets will be less but not trivial. Thus, we expect some divergence between the average and marginal costs.

Over the short term, subsidies will be necessary. Qualification procedures will most likely become more efficient and effective over time and, hence, less costly. We also believe that efforts should be made to encourage contributors to submit assets for qualification; subsidizing qualification is one form of encouragement.

Under the private property rights regime, the average costs minus a subsidy should be paid by the contributor of an asset when it is submitted for qualification. Over the longer term, these subsidies can be gradually withdrawn as greater numbers of assets are submitted.

e. General Reuse Training and Consulting

Training and consulting services will be needed both by contributors and by subscribers. Training for contributors will focus on how to design generalized, reusable components, and training for subscribers will focus on how to take advantage of reuse in building applications. These services may best be privatized in the long run. The key function of the repository would then be to provide a central location for advertising of training and consulting expertise.

There are some start-up (fixed) costs to course preparation, but most training cost accrue incrementally (staff time and materials); therefore, the marginal and average costs should be about the same.

During the start-up process, it may be appropriate to subsidize the original course development expense and, possibly, some early training expenses as a marketing device. The long-term goal is for the costs of all training and consulting services to be paid by the organizations receiving those services.

B. GOVERNMENT PROPERTY RIGHTS

In contrast to the simplicity of setting asset prices under private property rights, setting prices under government property rights is much more complicated.

1. Pricing Assets

a. Domain Models and Architectures

Under this scheme, domain models and architectures remain under the control of the government. Our recommendations for these are the same as those under private property rights (see Section V.A.1).

b. Source Code and Other Software Assets

In Chapter II, we distinguished between three categories of software assets that could reside in a repository. One category is existing software designed and implemented for a single project without reuse in mind. Such software that is turned over as is to the repository. For this category, no additional investment has been made to make the software more generalizable and, hence, more reusable. The second category is existing software that has been improved or "productized," usually to be resold for profit. The third category has software that is designed from the beginning to be general and reusable. In our view, under government property rights regime, all of the software assets will fall in the first category. We say this because the second and third categories require additional investment and the incentives for this type of investment exist only under the scenario of private property rights. The following discussion focuses on assets in the first category.

Though the software code may have been expensive to develop, all the development costs have already been incurred. Thus, the repository will not have to recover development costs per se. There are, of course, other costs associated with assets

(cataloging, searching), all of which will have to be either subsidized or paid by the subscribers because, under this scenario, contributors have no incentive to cover any costs.

2. Repository Services

The pricing considerations for the services provided by the repository under government property rights are explored here.

a. Basic Subscriber Services

The basic repository services of setting up user accounts and providing a help line are the same as under the scheme of private property rights (Section V.A.2.a).

b. Cataloging

Under the private property rights scenario, we recommended that the cost of cataloging be paid by the asset contributors. Under the government property rights scenario, no group of contributors are motivated to cover these costs. The cost of cataloging will have to be subsidized by the repository or paid by the subscribers. We recommend that the average cost be charged to the subscriber when an asset is downloaded.

c. Searching

Searching remains the same as under the scheme of private property rights (Section V.A.2.c).

d. Certification and Qualification

Pre-existing code that is provided to the repository is likely to be of uncertain quality. All current repositories suffer from this problem. Under the scheme of government property rights, certification becomes very important because (depending on what form it takes) it provides some assurance that the documentation is accurate and complete and that the asset compiles and executes. In our view, certification will be less important under private property rights because the contributors have strong incentives to provide high-quality assets.

Under both scenarios, qualification will be important for domain-specific assets.

Under the scenario of private property rights, we recommended that the contributor of an asset cover the costs for qualification. This will not be possible under the scenario of government property rights, because there are no contributors with an incentive to pay for

qualification (or certification). Under government property rights, the costs must be subsidized by the repository or paid by the subscribers.

e . General Reuse and Asset-Specific Training and Consulting

The need for general reuse training is the same here as under the scenario of private property rights (Section V.A.2.e). In some cases, training specific to particular assets will also be needed. (Under private ownership, this is a service that the contributor would provide.)

We recommend charging the average costs to the organization receiving the training. It may be appropriate to provide a subsidy in the beginning to cover the original course development.

f . Maintenance

Under government property rights, a decision must be made regarding asset maintenance. In the absence of additional investment to generalize them, most assets will require significant modification to tailor them to new applications. These modifications will be in addition to any changes necessary to correct defects or improve performance. The alternatives are to give the responsibility for maintenance either to the subscribers or to the repository. With the first alternative, the subscribers would be given the source code and allowed to make whatever changes they want. However, this approach results in severe configuration management and control problems (for an example, see Reference [12]). Such problems eliminate one of the major advantages of a repository.

With the second alternative, the repository would make all source code modifications (along with changes to the associated documentation). With the responsibility for maintenance comes the need to provide a mechanism for reporting problems as well as the need to provide configuration management (i.e., keeping track of which subscribers have which versions).

The problem with this alternative is that modifying the source code would likely prove to be very expensive and require a tremendous amount of application-specific expertise. If the subscribers take on the responsibility for maintenance, they will bear all costs directly. If the repository takes on this responsibility, the costs of tailoring assets to specific application systems can be substantial and should be paid for by the individual subscribers. Problem reporting and configuration management should have average and marginal costs that are similar. Thus, the subscribers should pay the average cost as part of an annual fee.

C. SUMMARY

Under a private property rights regime, the pricing for the repository is simplified because the contributors and subscribers set the prices for assets. The number and complexity of services offered is also simplified. Under the alternative regime of government property rights, specific recommendations for pricing repository services and assets were detailed. Our recommendations for pricing under both schemes are summarized in Chapter VI.

VI. SUMMARY AND RECOMMENDATIONS

Software repositories can play a critical role in fostering a market in reusable assets. Chapter II discusses the repository's technical role in managing assets and briefly describes current practices and procedures at several existing repositories. Chapter III discusses the repository's economic role. The repository serves to reduce the risk and uncertainty of both the contributor and subscriber of software products and services. This reduction of risk and uncertainty has the effect of reducing the costs to the contributors and subscribers. It also has the effect of allowing the economic actors to specialize and capture economies of scale in their operations. In turn, economies of scale have the beneficial effect of lowering unit costs. The government benefits from all this by receiving higher quality, lower cost, more maintainable software.

Chapter III also discusses how to set prices for repository products and services in order to strike an optimal balance between encouraging reuse and recovering costs. We noted several problems with trying to apply economic theory to repository pricing, the primary one being that repositories in many cases would not be able to recover their costs for assets while charging the optimal price. One possible solution could take the form of subsidies to the repository.

A preferable option, discussed in detail in Chapter IV, is to turn over most asset-pricing decisions to contributors and subscribers. In Chapter IV, we make the argument that existing code needs to be generalized in order to make reuse worthwhile to potential subscribers. We cited studies that point out the significant cost to generalizing code in order to make it reusable. We asked the question: Who has the incentive to incur these additional costs under the regime of government ownership? We believe the answer in most cases is "Nobody." The most likely outcome under the government property rights regime is a repository full of unusable and unused software assets.

We maintain that private property rights would create incentives to supply reusable assets. In addition, the private property rights regime eliminates the government's responsibility for dealing with a host of complicated problems associated with setting the price of repository assets and a few repository services. Under this regime, the repository would have fewer functions to perform than under the government ownership regime. The

functions we believe the repository would be required to perform and our pricing recommendations for these functions are presented in Table VI-1.

Table VI-1. Private Property Rights Pricing Recommendations

Repository Function	Pricing Recommendations
Basic Subscriber Services	The average cost should be paid by the subscribers in the form of a yearly subscription fee.
Cataloging	The average cost should be paid by the contributors when an asset is accepted into the repository.
Searching	The long-term goal is for all search costs to be paid by subscribers in the form of a commission when an asset is selected for use.
Qualification	The marginal costs should be paid by the contributors. Short-term subsidies may be needed.
General Reuse Training and Consulting	The marginal costs should be paid by the organizations receiving the services. Short-term subsidies are needed to cover initial course preparation.
Domain Models and Architecture	Free to subscribers and contributors.

For reasons delineated in the report, the DoD may finds it necessary to pursue government ownership of repository assets. In this case, the number and complexity of functions required by the repository are considerably larger than under the private property arrangement. Table VI-2 presents a summary of these functions and our pricing recommendations.

The DoD has a significant role to play in software reuse. Indeed the DoD's active participation is a necessary condition for success of the reuse initiative. The delineation and enforcement of domain models and architectures are obvious examples of functions that only the DoD can reasonably carry out. The DoD software repository also serves the vital role of bringing contributors and subscribers together.

But, we believe the DoD cannot satisfy the sufficient conditions for success alone. That requires sorting out the complex economic interactions among potential repository contributors and subscribers. The goal is to strike a balance between encouraging the diffusion of reusable assets and rewarding the asset creators for their efforts. This balance is best achieved when the economic actors are free to act in their own interests—not under the restraint of a pricing scheme imposed by some third party.

Table VI-2. Government Property Rights Pricing Recommendations

Functions	Same as Private Property Rights?	Pricing Recommendations
Basic Subscriber Services	Y	The average cost should be paid by the subscribers in the form of a yearly subscription fee.
Cataloging	N	The average cost should be paid by the subscriber when an asset is downloaded. Short-term subsidies may be necessary.
Searching	Y	The long-term goal is for all search costs to be paid by subscribers in the form of a commission when an asset is selected for use.
Certification and Qualification	N	The average cost should be paid by the subscribers. Short-term subsidies will be necessary.
General Reuse and Asset-Specific Training and Consulting	Y[a]	The marginal costs should be paid by the organizations receiving the services. Short-term subsidies are needed to cover initial course preparation.
Domain Models and Architecture	Y	Free to subscribers and contributors.
Maintenance	N	No charge if done by the subscriber. If done by the repository, all tailoring paid by the individual subscriber. Defect correction, problem reporting and configuration management paid by the subscribers in the form of an annual fee.
Source Code	N	The source code per se provided free (for associated services see entries above)

[a] Pricing is the same; repository would have to supply asset-specific training.

APPENDIX A

CURRENT PRACTICES AT THREE REPOSITORIES

APPENDIX A

CURRENT PRACTICES AT THREE REPOSITORIES

DEFENSE SOFTWARE REPOSITORY SYSTEM

The following information is based on a telephone conversation in January 1994 with Barbara Fleming. Ms. Fleming is the Technical Director, Software Systems Engineering Directorate, DISA/CIM.

Background Information

The DISA/CIM repository, known as the Defense Software Repository System (DSRS) is one of seven interconnected repositories (called "centers"). Along with DISA/CIM, the Army, Navy, Marine Corps, Air Force, Defense Logistics Agency, and National Security Agency each has a center. The subscribers are government employees and government contractors.

Assets across the seven centers total 8,800; they consist of source code and other products. This number includes a lot of duplication of assets. Most of the assets were developed under government contract, but some are commercial products.

Fees

Users are charged no fees. For assets owned by commercial vendors (e.g., Booch components), DISA/CIM's license permits distribution to any subscriber. The repository used to have a license for other components but the components became so expensive that the license was dropped. Now there is simply a pointer to those components, and the subscribers are told to contact the vendor.

Maintenance

At the DISA/CIM center, personnel keep track of each extraction and try to follow up with a survey to find out if an asset is actually used. They also notify users if a defect/problem with an asset has been reported.

Some, but not all of the assets are maintained. Some maintenance is done by the contributors. DISA/CIM maintains other assets via re-engineering.

Distribution

Most assets are stored on-line and distributed electronically via direct dial-up or Internet. In some cases, assets are stored off-line and are distributed on disk.

Data Rights

Everything in the repository is available to the government and government contractors. This availability is verified for each asset at the time it enters the repository. Some people would like to see all the assets put in the public domain. This requires DISA/CIM to determine the specific data rights associated with each asset, a difficult and time-consuming task. Consequently, only took a small percentage of the assets have been placed in the public domain.

DISA/CIM has learned to have all the contractual mechanisms in place to explicitly spell out the data rights associated with an asset when it is submitted to the repository. DISA/CIM is also considering a scheme wherein groups of assets will be accessible to specific groups of users. Thus, someone in the general public would be able to access only those assets in the public domain while a government employee or contractor would be able to access all assets in the repository.

Quality Control

All assets undergo some form of certification when they are placed in the repository. There are four levels of certification plus an "other" category. An example of other would be that training materials are included.

COSMIC

The following information is based on a telephone conversation in January 1994 with Scott Clark, Director of Marketing at COSMIC.

Background Information

NASA's COSMIC repository is an interesting case because it has been operating under a fee-for-service structure (which NASA refers to as "cost recovery") for years. The objective is to cover operating expenses, breaking even. COSMIC was founded in 1966 as a software repository. It offers source code and associated documentation. Most of the software represents entire programs rather than small utilities. COSMIC currently resides at the University of Georgia. Originally a totally free service, COSMIC began in the early 1970s charging subscribers $25,

which provided unlimited access. COSMIC now charges fees for individual assets. A fall 1993 COSMIC newsletter listed assets ranging in price from $100 to $7,000.

No fees are charged to NASA employees and NASA contractors. All assets are in the public domain, but others must pay the fee.

COSMIC currently has approximately 800 assets. Periodically, COSMIC culls out assets that do not appear to have a lot of reuse potential. COSMIC services several hundred users annually.

The majority of assets were paid for by NASA and developed by NASA Laboratories or contractors. Their submission is coordinated through the NASA centers and is required by a NASA-wide directive. A small number of assets come from other sources, including the Department of Defense and the Nuclear Regulatory Commission.

Fees

The repository sets the fee. COSMIC has one price for IBM-compatible personal computers and Macintoshes, one for workstations, and one for mainframes. Such factors as the size of the program and the number of different hardware platforms it runs on are also taken into account. The fees are used to cover the cost checking out the program and procuring the hardware needed to run it.

When an asset has been accepted into the repository, the authors receive a nominal payment (several hundred dollars).

Maintenance

COSMIC personnel have made a deliberate decision not to maintain the assets in the repository because they cannot be experts for all the different programs they offer, and the authors of the programs are in the best position to maintain them. When problems are found with an asset, the repository contacts the authors and tries to obtain either a patch or a new version. Patches are provided free (in the form of a letter to all users of the asset telling them where to make the change). New versions are provided for a moderate fee (usually one hundred dollars).

Distribution

Most assets are distributed via tape or diskette. Many of the users are not connected to COSMIC electronically.

Data Rights

All users of an asset are provided with the asset's source code and are free to make any changes.

Quality Control

All assets are compiled and executed on the platforms for which they are designed to run. COSMIC personnel also run test data-checking output. If test data are provided by the contributor, they are used. If not, test data are generated by COSMIC personnel. Personnel also make sure that the documentation is complete. They have found that the quality is variable and very much dependent on the particular center that coordinates the submission. There is currently no real incentive for the contributors except for the NASA directive

ASSET SOURCE FOR SOFTWARE ENGINEERING TECHNOLOGY

The following information is based on a telephone conversation with Karen Dennis. Ms. Dennis was then the Asset Source for Software Engineering Technology (ASSET) data librarian.

Background Information

ASSET is sponsored by DoD's Software Technology for Adaptable, Reliable Systems (STARS) program and is located in Morgantown, West Virginia. A large number of the assets come from the STARS program, but assets are actively solicited from any source. All assets are in the public domain and are provided free of charge. ASSET also provides pointers to commercial vendors. ASSET currently has approximately 300 assets.

Fees

ASSET currently charges no fees. There is a plan to change ASSET into a completely self-supporting entity within five years. Services are viewed as a definite source of income while some assets, at least, may continue to be provided at no charge. ASSET may eventually charge for serving as a broker between subscribers and commercial vendors.

Maintenance

ASSET currently does not maintain assets plans call for this service to be added. ASSET personnel do record problem reports and try to help the users whenever possible.

Distribution

Assets are distributed electronically.

Data Rights

All assets in the repository are in the public domain.

Quality Control

ASSET will eventually implement a four-level certification. Assets are currently checked to make sure they are complete and that they compile. Later certifications will include executing against test data.

APPENDIX B

AN EXAMPLE OF GENERALIZATION

APPENDIX B

AN EXAMPLE OF GENERALIZATION

We argue in Chapter V that software is made reusable by making it generalizable. Generalization involves removing some of the problem-specifics from a software component and requiring the user of the component to re-supply those details as parameters to describe the specific context of reuse.

Here, we provide an example of generalization to make the process clear. Take a simple single-purpose error message routine as shown below.

```
with Text_Io;
with Device_Names;
procedure Gyro_Spinup_Error is
begin
   Text_Io.Put_Line (Device_Names.Status_Recording,
                     "Error: The gyros are not up to speed.");
end Gyro_Spinup_Error;
```

In this simple example, the details needing classification for generalization are (1) the output device, (2) the Put_Line reference, (3) the error message string, and (4) its prefix ("Error:"). We assume that, of these four details, only the error message string and the output device are considered to be problem-specific details that do not belong in a generalized version of this component. In that case, a problem-independent generalization would be:

```
with Text_Io;
procedure Report_Error (Message : String;
                        On_Device : Text_Io.File_Type) is
begin
   Text_Io.Put_Line (On_Device, "Error: " & Message);
end Report_Error;
```

This example embodies less functionality but is more general, which is the usual tradeoff. Also, the removal of the problem-specific dependence on the package Device_Names means the unit will now compile in any context, another sign of its

increased generality. Because of the reduced function in the procedure, however, a call to the procedure must resupply what is missing. For example:

```
Report_Error ("The gyros are not up to speed.",
              Device_Names.Status_Recording);
```

would restore the specific functionality contained in the original example. The user of the routine must now make the decision of what actual parameters to bind to the formal parameters introduced in the more general example. More generality is possible but at the price of resupplying some of the detail from the point of reuse (the call statement).

REFERENCES

REFERENCES

[1] Department of Defense. "Department of Defense Software Technology Strategy," December 1991.

[2] Bailey, John W. "A Component Factory for Software Source Code Re-Engineering and Reuse," Ph.D. Dissertation, University of Maryland, 1992.

[3] Selby, Richard W., "Empirically Analyzing Software Reuse in a Production Environment" in *Software Reuse Engineering Technology.* Will Tracz (ed.), IEEE Computer Society Press, 1989.

[4] Frazier, Thomas P. "Software Reuse and Productivity: An Empirical View," in *Software Engineering Economics.* P. Gernier (ed.), Springer-Verlag, 1993.

[5] Cruickshank, Robert D., and Gaffney, John E. "An Economic Model of Software Reuse," in *Software Engineering Economics.* T. Gulledge and J. Lovelace (ed.), Springer-Verlag, 1992.

[6] Basili, Victor, R. and Barry T. Perricone. "Software Error and Complexity: An Empirical Investigation." *Communications of the ACM January*, 1984: 42-52.

[7] Basili, Victor R. "Metric Analysis and Data Validation Across FORTRAN Projects." *IEEE Transactions on Software Engineering.* SE-9, November 1983, pp. 652-663.

[8] Krueger, Charles W. "Software Reuse," *ACM Computing Surveys*, Volume 24, Number 2, June 1992.

[9] Department of Defense, Center for Software Reuse, "Fee-for-Service," 22 February 1993.

[10] Womer, N. K., and K. A. King. "Fee-for-Service Technical Letter," MITRE Corporation, 17 January 1992.

[11] Banker, Rajir D., Robert J. Kauffman, and Dani Zweig. "Repository Evaluation of Software Reuse." *IEEE Transactions on Software Engineering,* Vol. 19, No. 4, April 1993.

[12] Dikel, David, J. Rosoff, P. Kengor, J. Lichtenstein, G. DeMarco, and J. Wilson. *Software Reuse Case Study: TRILLIUM.* IBM Federal Systems Company, September 13, 1993.

[13] Barkema, Alan, and Michael L. Cook. "The Changing U.S. Pork Industry: A Dilemma for Public Policy." *Federal Reserve Bank of Kansas City Economic Review*, Second Quarter 1993.

[14] Ramsey, Frank P. "A Contribution to the Theory of Taxation." *Economic Journal*, 37, March 1927, pp. 47-61.

[15] *Proceedings from the Second Annual West Virginia Reuse Education and Training Workshop.* West Virginia University, October 1993, pp. 25-27, 117-125.

[16] Besen, Stanley M., and Raskind, Leo J. "An Introduction to the Law and Economics of Intellectual Property." *The Journal of Economic Perspectives*, Vol. 5, No. 1, Winter 1991, p. 3-27.

[17] Bator, Francis M. "The Anatomy of Market Failure." *Quarterly Journal of Economics*, August 1958.

[18] Office of the Director of Information Systems for Command, Control, Communications and Computers, "Army Strategic Software Reuse Plan." August 31, 1992.

[19] SoftTech, Incorporated. "Legal/Acquisition Issues: A Technical Report." Final Draft. Prepared for Department of Defense Center for Software Reuse Operations, February 1993.

[20] Samuelson, Pamela. "Toward a Reform of the Defense Department Software Acquisition Policy." SEI, CMU/SEI-86-TR-1 (NTIS ADA 169705), April 1986.

ABBREVIATIONS

ADP	automatic data processing
ASSET	Asset Source for Software Engineering Technology
CIM	Center for Information Management
DISA	Defense Information Systems Agency
DoD	Department of Defense
DSRS	Defense Software Repository System
STARS	Software Technology for Adaptable, Reliable Systems

REPORT DOCUMENTATION PAGE

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 2220-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE September 1994	3. REPORT TYPE AND DATES COVERED Final Report, May 1993 – Sep 1994

4. TITLE AND SUBTITLE

Economic Foundations for Pricing Software Reuse Repositories

5. FUNDING NUMBERS

MDA 903 89C 0003
T-J7-1158

6. AUTHOR(S)

Thomas P. Frazier, Elizabeth K. Bailey, and Bruce N. Angier

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Institute for Defense Analyses
1801 N. Beauregard Street
Alexandria, VA 22311-1772

8. PERFORMING ORGANIZATION REPORT NUMBER

IDA Paper P-2975

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

OASD(C3I)
1225 Jefferson Davis Highway, Suite 910
Arlington, VA 22202-4301

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12A. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12B. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Establishing and operating software repositories are important elements in the DoD's plan to encourage software reuse. One way to manage such repositories is through a fee-for-service arrangement, which involves charging a fee to the users of the repositories and using the proceeds to reimburse the repository for costs incurred in providing products and services. This paper examines the economic foundations for setting prices for repository products and services that strike an optimal balance between encouraging reuse and recovering costs. We argue that the most important issue in pricing is ownership of the assets. We believe that the most likely outcome under government ownership is a repository full of unusable and unused software assets, and maintain that private ownership would create incentives to supply truly reusable assets. In addition, the private ownership scheme eliminates the government's responsibility for dealing with the complicated problems associated with setting the price of repository assets and a few repository services.

14. SUBJECT TERMS

Software Reuse, Software Repositories, Cost Recovery, Costs, Setting (Adjusting), Economic Impact, Property Rights

15. NUMBER OF PAGES
58

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR